Assigning Your Application an Identity with Code Signing

Session 504

These are confidential sessions—please refrain from streaming, blogging, or taking pictures.

Assigning Your Application an Identity with Code Signing

Perry the Cynic
Lead Architect, OS Security Group

What's in It for You

- Recognize, filter, and organize groups of code
- System recognizes your software updates
- Resilience against unintended modifications
- Write first-class code
  - Unsigned code is legacy

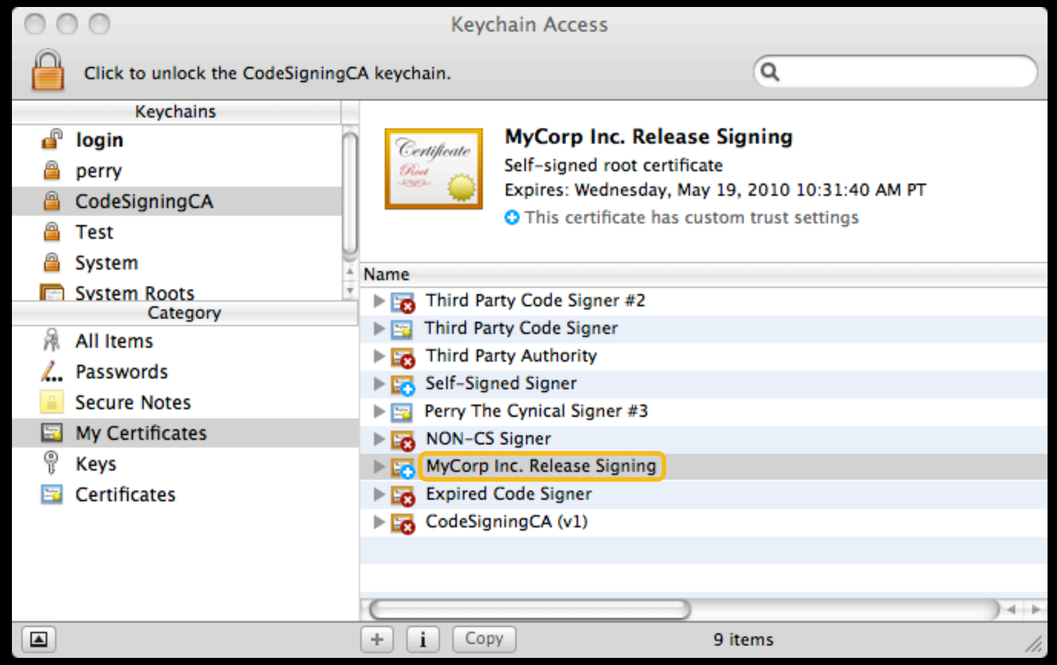
What Code Signing Does

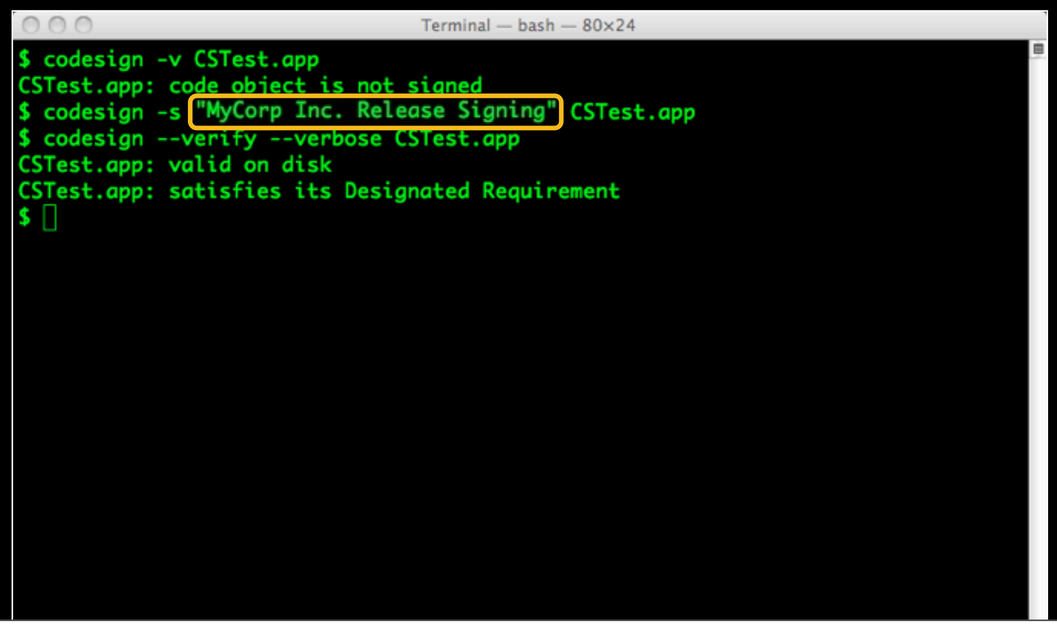
- Define identity for code
- Seal code to detect unintended alteration
- Provides a language for discussing identity
- Reliably identify running code

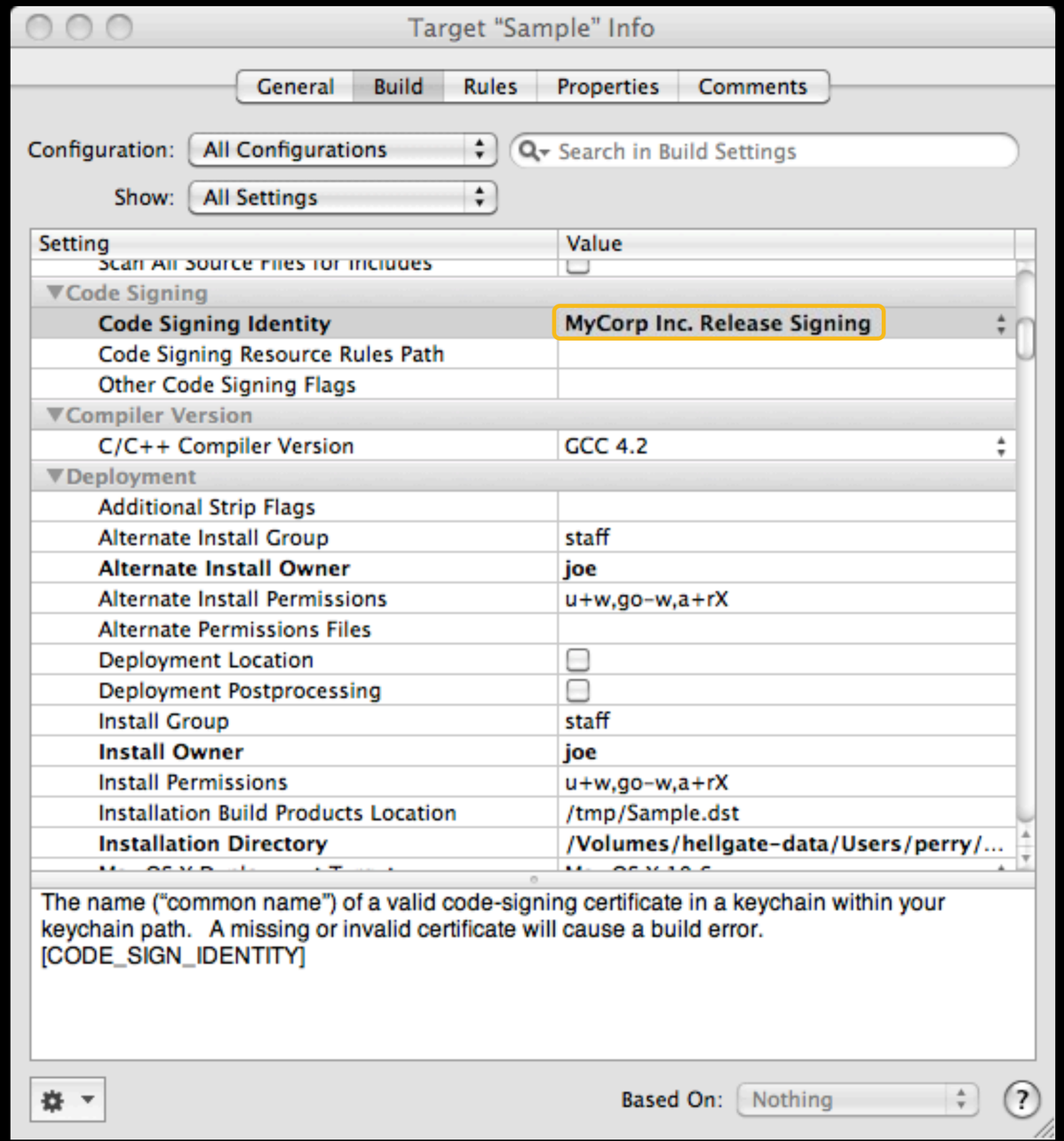
What Code Signing Does Not Do

- Provide privileges
- Protect against bugs
- Protect against misplaced trust

Signing Code in One Slide







Code Signing Is Pervasive

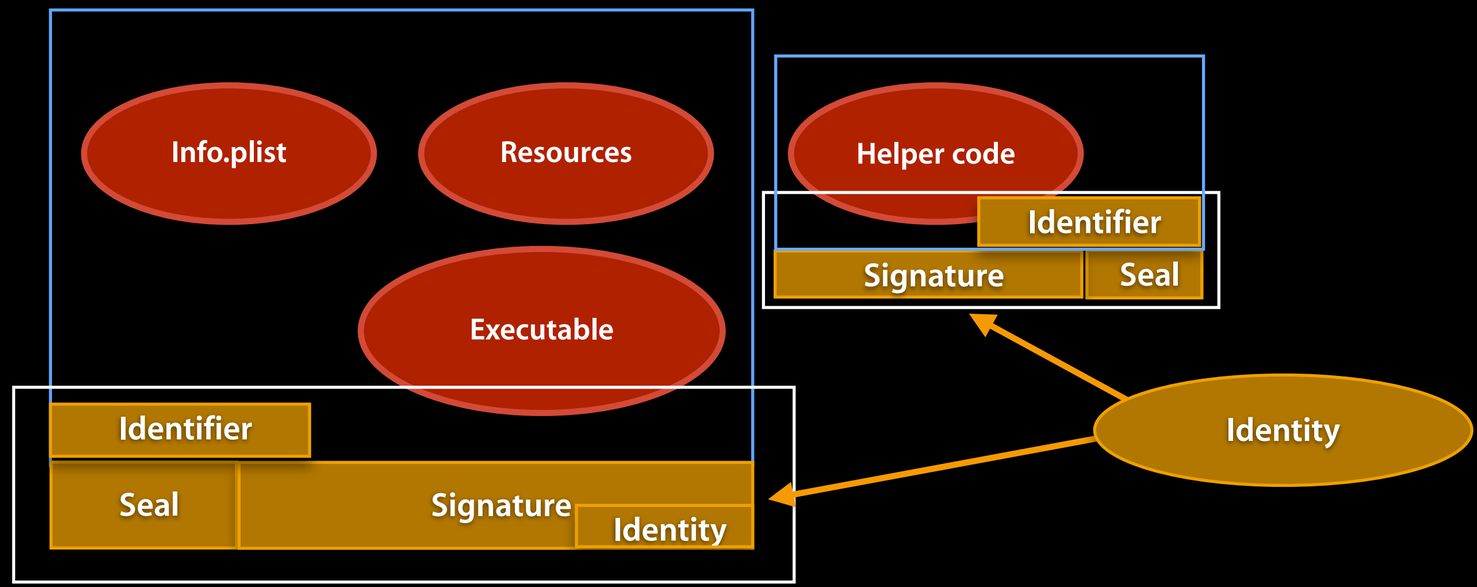
- All kinds of code
  - Programs
  - Libraries and plug-ins
  - Scripts
  - Bundles
  - Extensible
- Dynamic and static validation
- All shipping versions of Mac OS
- Goal: All code on the system is signed

This Is a Runtime Feature

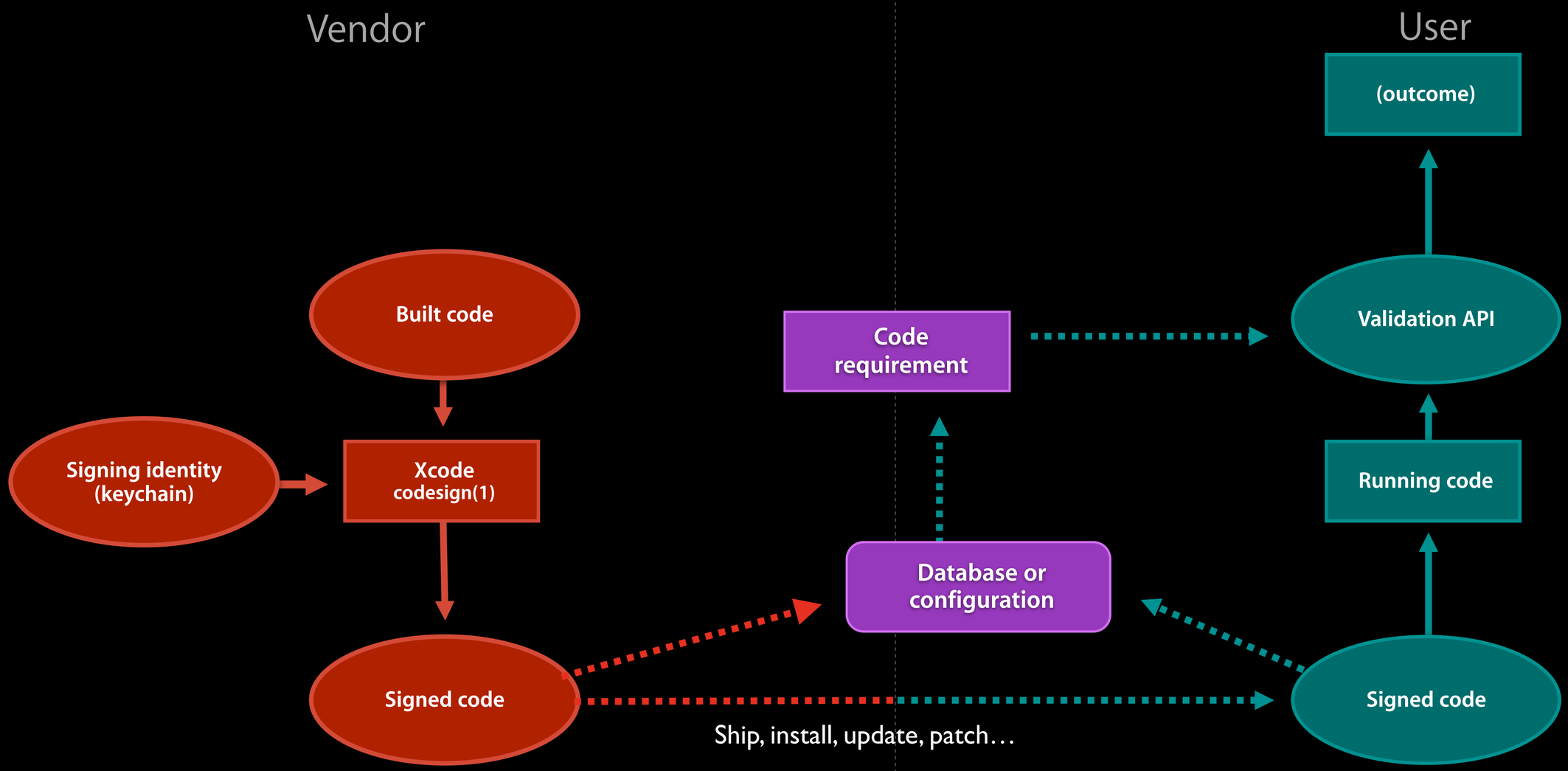
- Primary mission is validation of running code
  - Bind running code to its disk origin
  - Track dynamic validity state
  - Manage hierarchies of managed code (“code hosting”)
  - Put it all together efficiently
- Static validation is just a useful subfunction

Demo

How a Signature Is Built



How a Signature Is Used



Code Requirements

- Filter rules apply to signed code
- Matches features of the code or its signature
- Defines code identity
- Text form (viewing, editing)
- Binary form (storage, application)
- Internal Requirements embedded in signatures

Designated Requirements

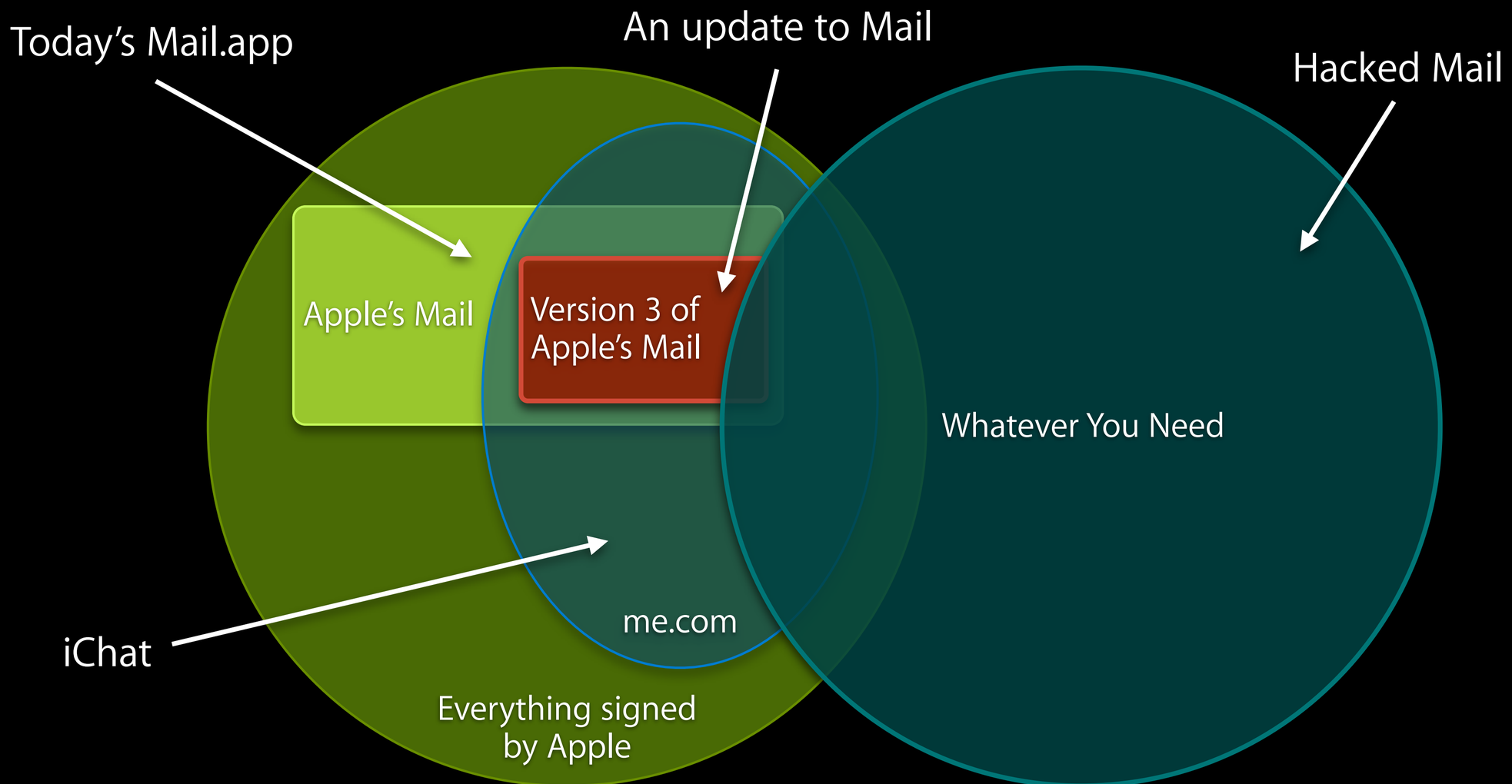
- Retrieved from signed code
- Implicitly generated or explicit by signer
- Explicit definition of “myself”
- Implicit definition of “update”

Code Identity

- Defined by requirement, not signature
- Classes of code (“all code that…”)
  - Even Designated Requirement (“all versions/updates of me”)
- Designated Requirement = self-identification
- One code answers to many identities
- Meaningless to compare signatures
- Code identity ≠ signing identity

Code Identity

(The Picture)

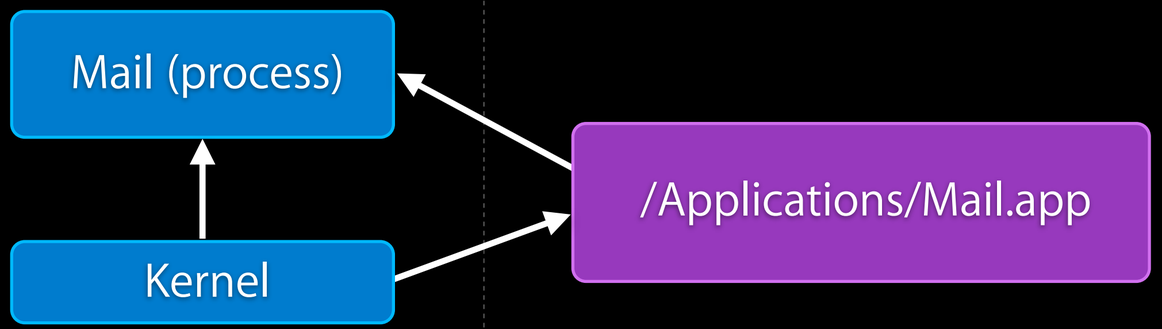


Code Hosting

Running code

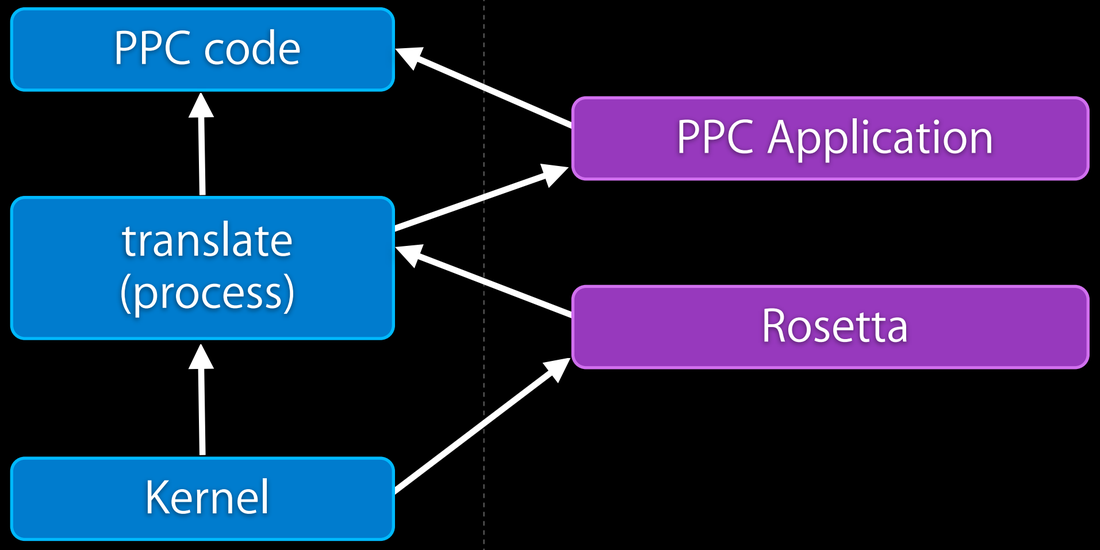
Static code

- Kernel hosts Mail.app



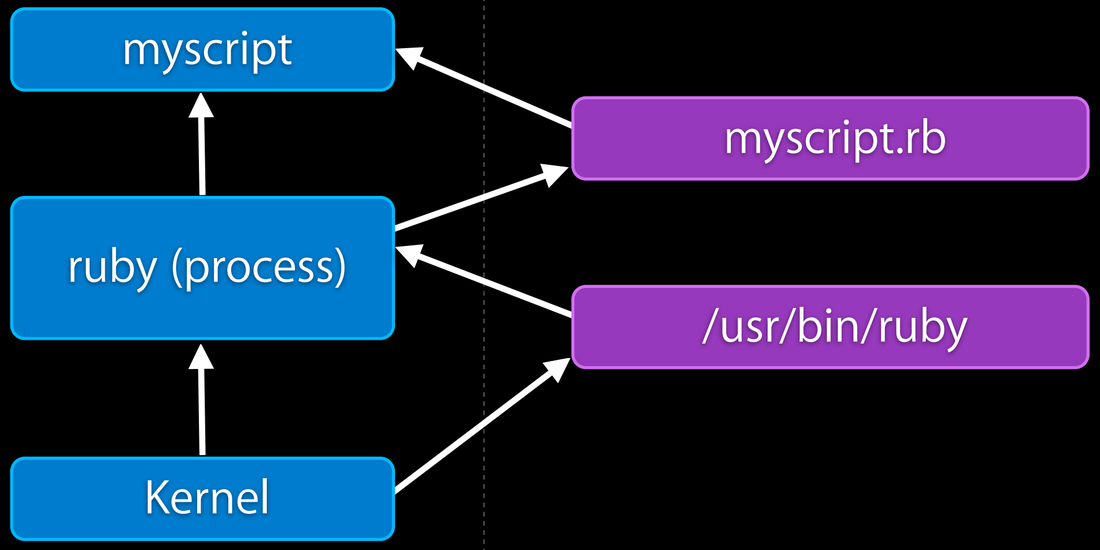
Code Hosting

- Kernel hosts Rosetta
- Rosetta hosts ppc application



Code Hosting

- Kernel hosts ruby interpreter
- ruby interpreter hosts ruby script

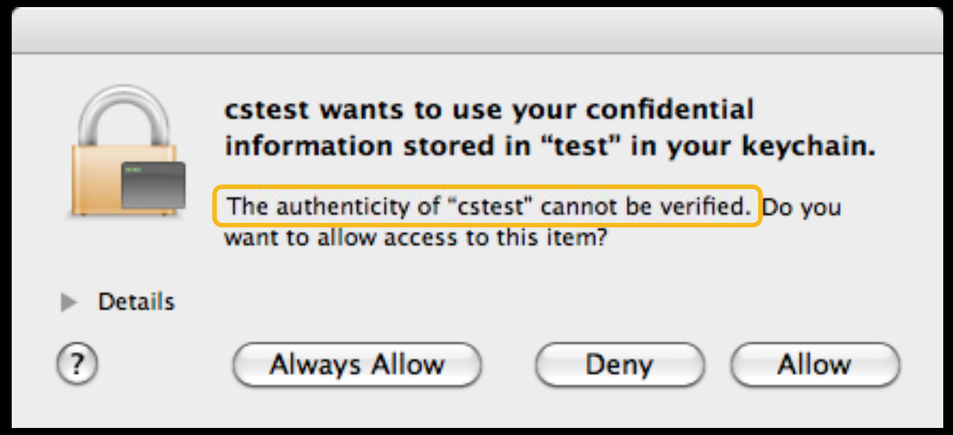


This could be
your customer's script

This could be Your program

What Happens to Unsigned Code?

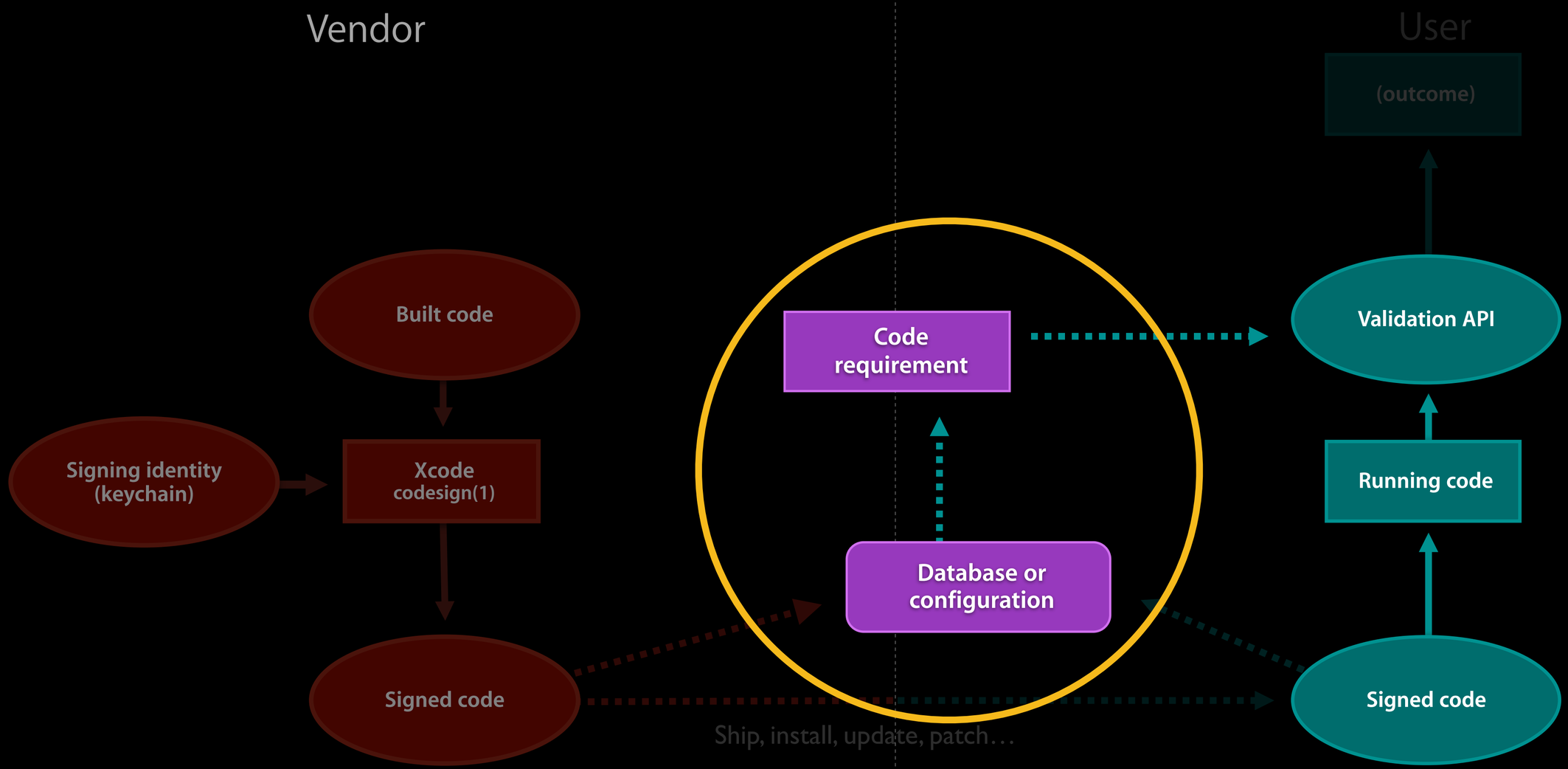
- Unsigned code has no reliable identity
  - Expect dialogs (some are critical)
- May be ad-hoc signed for you
- May be forced onto legacy path
  - Keychain dialogs, no equivalence tracking
- May be rejected outright
- Updates aren't recognized



What Happens if a Signature Breaks

- Broken code has no identity at all
  - Can’t trade on its identity
  - Can’t create keychain items
- No automatic repair
- Remedies
  - Statically invalid: Reinstall
  - Dynamically invalid: Relaunch (and don’t do that again)

Case Studies

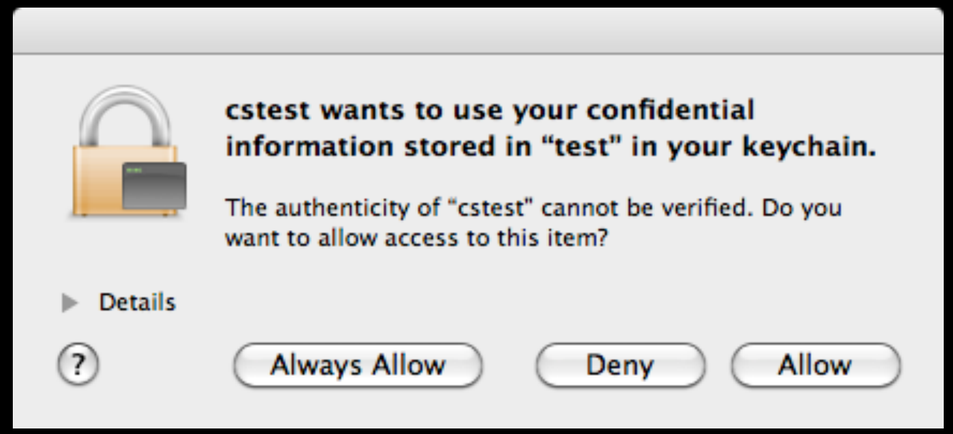


Case Study: Firewall

- Keep list of allowed applications
- Policy
  - Pre-approve system-trusted signatures
  - Prompt for others and remember
- Storage: system database

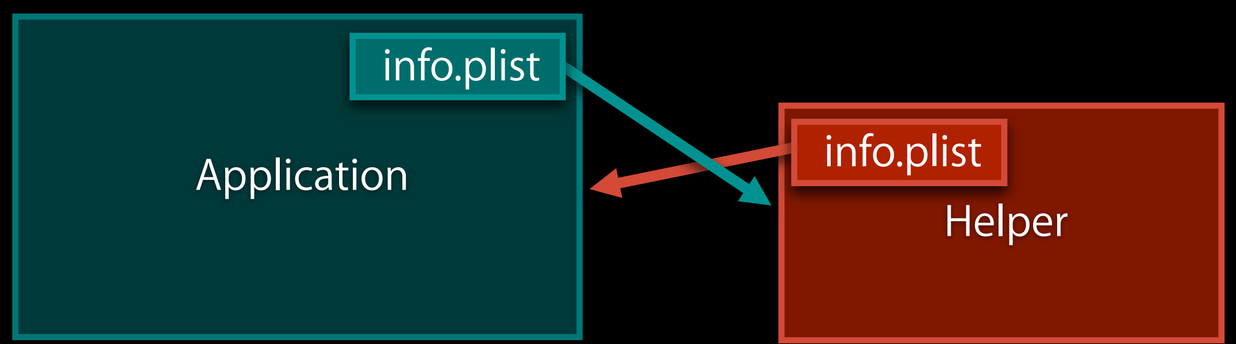
Case Study: Keychain ACLs

- Application list in each keychain item's ACL
- Policy: Explicit (list of code requirements)
  - From Designated Requirement of applications
  - Explicitly specified
- Storage: Embedded in ACL
  - Secured by keychain's security features



Case Study: Service Manager

- Application/Helper identify each other using code requirements
- Policy: Explicit
- Storage: In Info.plists



Signing Identities

- Industry-standard format (RFC 2459 et al)
- Sources
  - Make your own (Certificate Assistant, openssl, …)
  - Commercial CAs (Verisign, Thawte, …)
  - Apple (for iPhone development only)
- Stored as a Keychain Identity
  - Key + Certificate (“My Certificates”)
  - Transmit using PKCS12 (.p12) files

Ad-Hoc Signing

- Most specific identity possible (“just this program”)
- Automatically applied by system
  - Parental controls
  - Firewall
- Architecture specific(!)
- May modify your program
- Cannot track updates

API

- New for Snow Leopard
- Client API
  - Gate access controls by code identity
  - Validate integrity of requesting code
  - Remember code identity for later
- Hosting API
  - Describe and manage hosted code for system
  - Share your (host's) execution with its guests
- Signing API—not yet—use codesign(1)

API

- CoreFoundation Objects
  - SecCodeRef—Running code
  - SecStaticCodeRef—Code on disk
  - SecRequirementRef—Code requirements
- Code objects are local, representing global things
- Validation in each client space

How to Get into Trouble

- Put code into a bundle's Resources directory
- Change code after signing it
  - Post-build mastering
  - Resource files
  - Outsourced localization
- Selective installs
- Symbolic link trickery
- Self-modifying code

More Information

Craig Keithley
Technology Evangelist
keithley@apple.com

man codesign(1)

man csreq(1)

Documentation
http://developer.apple.com/documentation/MacOSX/Conceptual/CodeSigningGuide
http://developer.apple.com/technotes/tn2007/tn2206.html
https://developer.apple.com/snowleopard/library/documentation/Security/Reference/CodeSigningRef/Reference/reference.html
http://developer.apple.com/documentation/developertools/Conceptual/XcodeBuildSystem

Related Sessions

Managing User Privileges and Operations with Authorization Services	North Beach Wednesday 3:30PM
iPhone Security Best Practices	Mission Wednesday 5:00PM
Managing Processes with launchd	Russian Hill Friday 2:00PM

Labs

Mac OS X Security Lab	OS X Lab A Wednesday 12:00PM

Summary

- Code Signing is about identity of code
- You define your program's identity
- The end-system's user/administrator makes the decisions
- Sign your code now
  - If you ship unsigned code, you are late

Q&A

WWDC Session Survey

You can rate your sessions on the WWDC Attendee Site.

Survey forms are located on each session description page at:

developer.apple.com/wwdc/attendee

WWDC09